Permutation_test

December 18, 2023

[215]:
```
from scipy import stats
import numpy as np
import random

def mystatistics1(*data):
    return (np.mean(data[0]) - np.mean(data[1]))

def mystatistics2(*data):
    return abs(np.mean(data[0]) - np.mean(data[1]))
```

0.1 Estimating the data from Amazon histogram

[211]:
```
np.array([.63,.18,.07,.05,.07])*257
```

[211]:
```
array([161.91,  46.26,  17.99,  12.85,  17.99])
```

[212]:
```
black = [5]*162 + [4]*46 + [3]*18 + [2]*13 + [1]*18; len(black)
```

[212]:
```
257
```

[213]:
```
np.array([.55,.16,.13,.07,.1])*114
```

[213]:
```
array([62.7 , 18.24, 14.82,  7.98, 11.4 ])
```

[214]:
```
blue = [5]*63 + [4]*18 + [3]*15 + [2]*8 + [1]*11; len(blue)
```

[214]:
```
115
```

0.2 Running the permutation test with 1- and 2-sided version

[216]:
```
stats.permutation_test([black, blue], statistic=mystatistics1)
```

[216]:
```
PermutationTestResult(statistic=0.2577228895279986, pvalue=0.082,
null_distribution=array([-0.1324649 , -0.1450516 ,  0.10668246, …,
0.08150905,
       -0.00659787, -0.04435798]))
```

[217]:
```
stats.permutation_test([black, blue], statistic=mystatistics2)
```

[217]:
```
PermutationTestResult(statistic=0.2577228895279986, pvalue=0.1474,
null_distribution=array([0.11987819, 0.03177127, 0.03116224, …, 0.05694468,
0.10668246,
       0.08211808]))
```

0.3 Fake data: what if we 2x the group of customers who bought blue?

Assuming the same percentages of each mark.

[218]:
```
stats.permutation_test([black, blue+blue], statistic=mystatistics2)
```

[218]:
```
PermutationTestResult(statistic=0.2577228895279986, pvalue=0.0548,
null_distribution=array([0.08470648, 0.01879547, 0.20005075, …, 0.05535442,
0.15885637,
       0.12126544]))
```

0.4 Fake data: what if we take random subgroup of customers who bought black?

Random sample => should have approax. the same average.

[219]:
```
small_black = random.sample(black, k=115)
np.mean(black),np.mean(small_black)
```

[219]:
```
(4.249027237354086, 4.252173913043478)
```

[220]:
```
stats.permutation_test([small_black, blue], statistic=mystatistics2)
```

[220]:
```
PermutationTestResult(statistic=0.26086956521739113, pvalue=0.2962,
null_distribution=array([0.12173913, 0.22608696, 0.08695652, …, 0.15652174,
0.20869565,
       0.36521739]))
```

[ ]:
```

```

[ ]:
```

```

[74]:
```
stats.permutation_test([[1,1],[2,3]], statistic=mystatistics)
```

[74]:
```
PermutationTestResult(statistic=1.5, pvalue=0.6666666666666666,
null_distribution=array([1.5, 0.5, 0.5, 0.5, 0.5, 1.5]))
```

[80]:
```
stats.permutation_test([[1,1],[2,3]], statistic=mystatistics,␣
↪alternative='less')
```

[80]:
```
PermutationTestResult(statistic=-1.5, pvalue=0.16666666666666666,
null_distribution=array([-1.5, -0.5,  0.5, -0.5,  0.5,  1.5]))
```

[84]:
```
stats.permutation_test([[1,1],[2,3]], statistic=mystatistics,␣
↪alternative='two-sided')
```

[84]:
```
PermutationTestResult(statistic=-1.5, pvalue=0.3333333333333333,
null_distribution=array([-1.5, -0.5,  0.5, -0.5,  0.5,  1.5]))
```

[85]:
```
stats.permutation_test([[1,1],[2,3]], statistic=mystatistics2,␣
↪alternative='greater')
```

[85]:
```
PermutationTestResult(statistic=1.5, pvalue=0.3333333333333333,
null_distribution=array([1.5, 0.5, 0.5, 0.5, 0.5, 1.5]))
```

[ ]:
```

```

[37]:
```
testa = [5]*5
testb = [4]*5
```

[38]:
```
stats.permutation_test([testa,testb], statistic=mystatistics) #,␣
↪random_state=211)
```

[38]:
```
PermutationTestResult(statistic=1.0, pvalue=0.015873015873015872,
null_distribution=array([1. , 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,
0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.6, 0.6, 0.6, 1. ]))
```

[60]:
```
stats.permutation_test([testa,testb], statistic=mystatistics, n_resamples=100)␣
↪#, random_state=211)
```

[60]:
```
PermutationTestResult(statistic=1.0, pvalue=0.039603960396039604,
null_distribution=array([0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6,
0.2, 0.2,
       0.2, 0.2, 0.6, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 1. , 0.2,
       0.6, 0.2, 0.2, 0.6, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.6, 0.2, 0.6, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.6, 0.2, 0.2, 0.2, 0.2, 0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.2, 0.2, 0.2, 0.6,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2])
```

[61]:
```
len([1. , 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.2, 0.2, 0.2,
       0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.2, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6,
       0.6, 0.6, 0.6, 0.6, 1. ])
```

[61]:
```
252
```

[ ]:
```

```

[67]:
```
testc = [5]*50 + [4]*40
testd = [5]*40 + [4]*50
```

[71]:
```
testc = [5]*500 + [4]*400
testd = [5]*400 + [4]*500
```

[72]:
```
stats.permutation_test([testc,testd], statistic=mystatistics) #,␣
↪random_state=211)
```

[72]:
```
PermutationTestResult(statistic=0.11111111111111072, pvalue=0.0002,
null_distribution=array([0.00222222, 0.00444444, 0.00444444, …, 0.02666667,
0.02666667,
       0.01555556]))
```

[ ]:
```
```

[22]:
```
amazon_active_black = ([5]*61 + [4]*20 + [3]*7 + [2]*6 + [1]*6)*6
len(amazon_active_black), mean(amazon_active_black)
```

[22]:
```
(600, 4.24)
```

[5]:
```
amazon_pro_tit = ([5]*53 + [4]*17 + [3]*9 + [2]*8 + [1]*12)*11
len(amazon_pro_tit), mean(amazon_pro_tit)
```

[5]:
```
(1089, 3.919191919191919)
```

[6]:
```
amazon_pro_beige = ([5]*61 + [4]*14 + [3]*9 + [2]*5 + [1]*10)*2
len(amazon_pro_beige), mean(amazon_pro_beige)
```

[6]:
```
(198, 4.121212121212121)
```

[8]:
```
amazon_active_mint = ([5]*59 + [4]*19 + [3]*9 + [2]*5 + [1]*8)*2
len(amazon_active_mint)
```

[8]:
```
200
```

[9]:
```
stats.permutation_test([amazon_active_mint,amazon_pro_tit],␣
↪statistic=mystatistics)
```

[9]:
```
PermutationTestResult(statistic=0.24080808080808103, pvalue=0.0522,
null_distribution=array([0.08693297, 0.05142332, 0.06325987, …, 0.12836088,
0.08693297,
       0.01959596]))
```

[136]:
```
stats.permutation_test([amazon_pro_beige,amazon_pro_tit],␣
↪statistic=mystatistics)
```

[136]:
```
PermutationTestResult(statistic=0.202020202020202, pvalue=0.1392,
null_distribution=array([0.28741965, 0.03076217, 0.11248852, …, 0.13820018,
0.13820018,
       0.07254362]))
```

[ ]:
```
```

[135]:
```
stats.permutation_test([amazon_active_mint,amazon_active_black],␣
↪statistic=mystatistics)
```

[135]:
```
PermutationTestResult(statistic=0.08000000000000007, pvalue=0.88,
null_distribution=array([0.09333333, 0.03333333, 0.06      , …, 0.07333333,
0.1       ,
       0.04      ]))
```

[ ]:

[112]:
```
stats.permutation_test([[1,1],[2,2]], statistic=mystatistics)
```

[112]:
```
PermutationTestResult(statistic=1.0, pvalue=0.6666666666666666,
null_distribution=array([1., 0., 0., 0., 0., 1.]))
```

[ ]:

1 Wilcoxon test

[55]:
```
stats.wilcoxon([1,3,4,-2])
```

[55]:
```
WilcoxonResult(statistic=2.0, pvalue=0.375)
```

[56]:
```
stats.wilcoxon([1,3,4,-2], alternative="greater")
```

[56]:
```
WilcoxonResult(statistic=8.0, pvalue=0.1875)
```

[57]:
```
stats.wilcoxon([1,3,4,-2], alternative="less")
```

[57]:
```
WilcoxonResult(statistic=8.0, pvalue=0.875)
```

[ ]:

[145]:
```
stats.wilcoxon([1,-2,3,4,5,6,7])
```

[145]:
```
WilcoxonResult(statistic=2.0, pvalue=0.046875)
```

[ ]:

[49]:
```
stats.wilcoxon([-1,3,4,2])
```

[49]:
```
WilcoxonResult(statistic=1.0, pvalue=0.25)
```

[50]:
```
stats.wilcoxon([1,3,4,-2])
```

[50]:
```
WilcoxonResult(statistic=2.0, pvalue=0.375)
```

[51]:
```
stats.wilcoxon([1,-3,4,2])
```

[51]:
```
WilcoxonResult(statistic=3.0, pvalue=0.625)
```

[21]:
```
stats.wilcoxon([1,3,-4,2])
```

[21]:
```
WilcoxonResult(statistic=4.0, pvalue=0.875)
```

[54]:
```
stats.wilcoxon([1,-3,4,-2])
```

[54]:
```
WilcoxonResult(statistic=5.0, pvalue=1.0)
```

2 Sign-test

[137]:
```
stats.binom.cdf(1,4,p=.5)*2
```

[137]:
```
0.625
```

[ ]:
```

```

[146]:
```
stats.binomtest(2,4,p=.5, alternative='two-sided')
```

[146]:
```
BinomTestResult(k=2, n=4, alternative='two-sided', statistic=0.5, pvalue=1.0)
```

[138]:
```
stats.binomtest(3,4,p=.5, alternative='two-sided')
```

[138]:
```
BinomTestResult(k=3, n=4, alternative='two-sided', statistic=0.75, pvalue=0.625)
```

[42]:
```
stats.binomtest(4,4,p=.5)
```

[42]:
```
BinomTestResult(k=4, n=4, alternative='two-sided', statistic=1.0, pvalue=0.125)
```

[32]:
```
5/16.
```

[32]:
```
0.3125
```

[40]:
```
stats.binomtest(3,4, alternative='greater')
```

[40]:
```
BinomTestResult(k=3, n=4, alternative='greater', statistic=0.75, pvalue=0.3125)
```

[41]:
```
stats.binomtest(4,4, alternative='greater')
```

[41]:
```
BinomTestResult(k=4, n=4, alternative='greater', statistic=1.0, pvalue=0.0625)
```

[140]:
```
1/16.
```

[140]:
```
0.0625
```

[139]:
```
stats.binomtest(3,4, alternative='less')
```

[139]:
```
BinomTestResult(k=3, n=4, alternative='less', statistic=0.75, pvalue=0.9375)
```

3 2

[160]:
```
import numpy as np
```

[161]:
```
with_sample_problems = np.array([591, 621, 683, 579, 451, 680, 691, 769, 563,␣
↪575])
without_sample_problems = np.array([509, 540, 688, 502, 424, 683, 568, 748,␣
↪530, 524])
```

[162]:
```
print(with_sample_problems-without_sample_problems)
```

```
[ 82  81  -5  77  27  -3 123  21  33  51]
```

[165]:
```
stats.wilcoxon(with_sample_problems, without_sample_problems,␣
↪alternative='less')
```

[165]:
```
WilcoxonResult(statistic=52.0, pvalue=0.9970703125)
```

[111]:
```
stats.wilcoxon(with_sample_problems-without_sample_problems,␣
↪alternative='greater')
```

[111]:
```
WilcoxonResult(statistic=52.0, pvalue=0.0048828125)
```

[75]:
```
np.size(with_sample_problems)
```

[75]:
```
10
```

[112]:
```
np.mean(with_sample_problems), np.mean(without_sample_problems)
```

[112]:
```
(620.3, 571.6)
```

[ ]:
```

```

[ ]:
```

```

[166]:
```
stats.wilcoxon(with_sample_problems, without_sample_problems+20,␣
↪alternative='greater')
```

[166]:
```
WilcoxonResult(statistic=46.0, pvalue=0.0322265625)
```

[167]:
```
stats.wilcoxon(with_sample_problems-without_sample_problems-20,␣
↪alternative='less')
```

[167]:
```
WilcoxonResult(statistic=46.0, pvalue=0.9755859375)
```

[156]:
```
stats.wilcoxon(with_sample_problems-without_sample_problems-50,␣
↪alternative='less')
```

[156]:
```
WilcoxonResult(statistic=28.0, pvalue=0.5390625)
```

[ ]:
```

```

[157]:
```
with_sample_problems-without_sample_problems-50
```

[157]:
```
array([ 32,  31, -55,  27, -23, -53,  73, -29, -17,   1])
```

[115]:
```
with_sample_problems-without_sample_problems>=20
```

[115]:
```
array([ True,  True, False,  True,  True, False,  True,  True,  True,
        True])
```

[159]:
```
np.sum(with_sample_problems-without_sample_problems>=20)
```

[159]:
```
8
```

[117]:
```
stats.binomtest(8, 10, alternative='greater')
```

[117]:
```
BinomTestResult(k=8, n=10, alternative='greater', statistic=0.8,
pvalue=0.0546875)
```

[148]:
```
X = list(with_sample_problems)
Y = list(without_sample_problems+20)
```

[169]:
```
X
```

[169]:
```
[591, 621, 683, 579, 451, 680, 691, 769, 563, 575]
```

[170]:
```
Y
```

[170]:
```
[529, 560, 708, 522, 444, 703, 588, 768, 550, 544]
```

[168]:
```
stats.permutation_test([X,Y],  statistic=mystatistics)
```

[168]:
```
PermutationTestResult(statistic=28.699999999999932, pvalue=0.4906,
null_distribution=array([ 39.1,   0.7,  -5.5, …, -95.9,  -5.3,  38.9]))
```

[171]:
```
stats.permutation_test([X*8,Y*8],  statistic=mystatistics)
```

[171]:
```
PermutationTestResult(statistic=28.699999999999932, pvalue=0.0538,
null_distribution=array([-14.175,  -0.675,  14.825, …,  18.95 ,   0.775,  22.4
```

```
]))
```

[ ]:

[ ]:

[ ]:

[175]:
```
a = np.array([531, 621, 663, 579, 451, 660, 591, 719, 543, 575])
b = np.array([509, 540, 688, 502, 424, 683, 568, 748, 530, 524])
```

[176]:
```
a.mean(), b.mean()
```

[176]:
```
(593.3, 571.6)
```

[179]:
```
np.sum(a-b>=50)
```

[179]:
```
3
```

[180]:
```
stats.binomtest(3,10,.5,alternative='less')
```

[180]:
```
BinomTestResult(k=3, n=10, alternative='less', statistic=0.3, pvalue=0.171875)
```

[ ]: